На правах рукописи

Степанов Евгений Александрович

Методы и средства планирования вычислений в системах автоматизированного динамического распараллеливания программ

Специальность 05.13.11 — математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Автореферат
диссертации на соискание ученой степени
кандидата физико-математических наук

Москва – 2007

Работа выполнена на механико-математическом факультете и в Институте механики Московского государственного университета имени М. В. Ломоносова.

Научный руководитель:	доктор физико-математических наук, профессор *Васенин Валерий Александрович.*
Официальные оппоненты:	доктор физико-математических наук, профессор *Кузюрин Николай Николаевич*; кандидат физико-математических наук, доцент *Пронкин Юрий Николаевич.*
Ведущая организация:	Федеральное государственное унитарное предприятие НИИ «Квант».

Защита диссертации состоится « 15 » февраля 2008 г. в 15 часов на заседании диссертационного совета Д.002.087.01 в Институте системного программирования Российской академии наук по адресу: 109004, Москва, ул. Большая Коммунистическая, д. 25.

С диссертацией можно ознакомиться в библиотеке Института системного программирования РАН.

Автореферат разослан « 14 » января 2008 г.

Ученый секретарь
диссертационного совета
канд. физ.-мат. наук — С. П. Прохоров

Общая характеристика работы

Актуальность темы

В последние годы наблюдается значительный рост производительности вычислительных систем различной архитектуры, от сильносвязанных суперкомпьютерных установок до распределенных слабосвязанных систем, создаваемых по технологии GRID. Для решения многих практически важных, и, как правило, вычислительно сложных задач применяются *параллельные программы*, исполняемые на многопроцессорных установках.

Одним из наиболее популярных средств для создания параллельных программ в настоящее время является стандарт MPI. Однако он обладает недостатками, затрудняющими написание программ на основе алгоритмов, в которых структура вычислительного графа, время исполнения отдельных частей и зависимости между ними неизвестны на стадии написания программы. Назовем такие приложения и алгоритмы обладающими *внутренним динамическим параллелизмом* (или, более кратко — *динамическим параллелизмом*). Значительные трудности возникают при использовании MPI-программ на существенно неоднородной вычислительной среде. Стандарт MPI определяет такие низкоуровневые операции, как отправка и прием сообщений с отдельного узла, а также пересылка сообщений и синхронизация процессов взаимодействия между группой узлов. Применение MPI позволяет создавать эффективные программы для решения многих прикладных задач, в первую очередь — обладающих внутренним статическим параллелизмом. В то же время, в стандарте MPI не описаны средства динамической балансировки вычислительной нагрузки. При использовании MPI для приложений с динамическим параллелизмом эти механизмы должны быть реализованы в приложении.

Одним из альтернативных подходов к созданию параллельных программ является их *автоматизированное динамическое распараллелива-*

ние[1]. При его использовании для приложений с динамическим параллелизмом передача данных, синхронизация процессов и распределение нагрузки выполняются автоматически, без указаний со стороны пользователя. Автоматизированное динамическое распараллеливание существенно сокращает время, которое требуется для реализации алгоритма. Оно уменьшает трудоемкость разработки для многих классов приложений. К таким, в первую очередь, относятся приложения, в том числе отмеченные выше, в которых на стадии написания программы распределить нагрузку между узлами вычислительного поля невозможно или очень сложно. К их числу относятся, например, задачи, сводящиеся к поиску слабоструктурировнных данных с использованием графовых моделей, или игровые задачи со сложными стратегиями.

В последнее время для решения вычислительно сложных задач широкое применение получают территориально распределенные высокопроизводительные вычислительные установки. Такие вычислительные комплексы характеризуются неоднородностью аппаратного и программного обеспечения, различной производительностью узлов и коммуникационных каналов, и, как следствие, высокой вероятностью отказов. Как показывает практика, использование традиционных методов статического распараллеливания на таких установках неэффективно.

Под *динамическим распараллеливанием* в контексте настоящей работы понимается способ создания параллельных программ, при котором распределение вычислений между узлами системы происходит в течение всего времени исполнения программы. Динамическое распараллеливание позволяет преодолеть недостатки, присущие традиционным методам распараллеливания в распределенных системах.

Ключевым компонентом в средствах динамического распараллелива-

[1] *Васенин В. А., Водомеров А. Н., Инюхин А. В.* Средства автоматизированного динамического распараллеливания программ на основе сочетания императивных и функциональных механизмов // *Информационные Технологии.* — 2007. — № 5. — 32 с.

ния является модуль планирования задач (далее именуемый *планировщиком*). Под планированием будем понимать процесс распределения вычислительной работы[2] между узлами с целью уменьшения времени исполнения программы.

К настоящему времени разработано несколько средств автоматизированного динамического распараллеливания вычислительных приложений. Многие из них, например, GUM и MultiLisp, работают с программами на функциональных языках. Программный комплекс NewTS является средством автоматизированного динамического распараллеливания программ, основанным на языках C и C++. Использование в качестве базовых широко распространенных, императивных языков программирования позволяет создавать с его помощью высокопроизводительные параллельные программы для задач с динамическим параллелизмом. Такой подход облегчает перенос существующих последовательных программ на вычислительные комплексы с новой архитектурой. Важной особенностью NewTS является тот факт, что программы для него могут исполняться как на локальных, сильносвязанных установках, так и на распределенных, в том числе гетерогенных, вычислительных кластерах. Последнее обстоятельство позволяет использовать NewTS в качестве удобного объекта для тестирования и апробации новых подходов к планированию вычислений на комплексах с разной архитектурой. Вместе с тем, планировщик, который использовался в первых версиях NewTS, обладал рядом существенных недостатков, затруднявших его применение на системах с большим числом вычислительных узлов, а также на гетерогенной вычислительной среде.

В силу изложенных выше причин, разработка алгоритмов планирования для средств автоматизированного динамического распараллеливания приложений, способных эффективно работать в условиях распределенной

[2]Под *вычислительной работой* здесь будем понимать процесс вычислений, обеспечивающих решение одной из подзадач в рамках исходной прикладной программы.

гетерогенной среды, а также их программная реализация в программном комплексе NewTS, составляющие суть диссертационной работы, являются актуальными задачами.

Цели и задачи работы

Целью диссертационной работы является *разработка методов и средств планирования в системах автоматизированного динамического распараллеливания приложений и их реализация в составе программного комплекса NewTS*. Для достижения этой цели были поставлены следующие задачи:

1. математическое описание процессов исполнения и планирования параллельных вычислительных приложений;
2. исследование свойств и разработка эффективных алгоритмов планирования процессов выполнения параллельных приложений с динамическим параллелизмом;
3. реализация разработанных алгоритмов в составе модуля планирования системы автоматизированного динамического распараллеливания программ NewTS;
4. исследование и разработка методов эффективного и корректного исполнения мелкозернистых параллельных программ.

Научная новизна работы

Научной новизной обладают следующие результаты диссертации:

- математическая модель, описывающая процесс исполнения параллельной программы в распределенных и неоднородных системах, и оценка эффективности исполнения параллельных программ;

- два алгоритма планирования для системы автоматизированного динамического распараллеливания приложений, реализованные в NewTS, которые позволяют повысить утилизацию ресурсов вычислительных систем и уменьшить время исполнения параллельных программ.

Практическая значимость

Разработанные алгоритмы позволяют эффективно исполнять широкий класс параллельных вычислительных программ. Созданная автором программная реализация этих алгоритмов для системы NewTS была применена для распараллеливания ряда практически значимых вычислительных задач, в том числе:

- программного комплекса Vortex, предназначенного для моделирования двумерного нестационарного обтекания твердых тел потоком несжимаемой среды;
- приложения RT, используемого для построения высококачественных изображений методом трассировки лучей;
- приложения insert_doc, для обработки и индексирования текстовых документов, входящего в состав поисковой системы АСИО[3].

Результаты применения подтвердили эффективность и масштабируемость разработанных алгоритмов.

Доклады и печатные публикации

Основные положения диссертации докладывались на международных научных конференциях студентов, аспирантов и молодых ученых «Ломоносов-2005», «Ломоносов-2006» и «Ломоносов-2007», на третьей международной

[3]АСИО — Автоматизированная Система Информационного Обеспечения, разрабатываемая в НИИ Механики МГУ имени М. В. Ломоносова для тематической обработки текстовой информации в больших (корпоративного масштаба) и сверхбольших (Интернет) хранилищах данных.

конференции по проблемам управления МКПУ-2006 (Москва, ИПУ РАН, 20–22 июня 2006 года), на IX международной конференции «Проблемы функционирования информационных сетей» ПФИС-2006 (Новосибирск, 31 июля – 3 августа 2006 года), а также на семинаре «Проблемы современных информационно-вычислительных систем» под руководством д.ф.-м.н., проф. В. А. Васенина (два доклада в течении 2005–2007 г. г.).

По материалам диссертации опубликовано семь работ, две из которых — в журналах, рекомендованных ВАК.

Структура работы

Работа состоит из введения, пяти глав, заключения и списка литературы. Общий объем диссертации — 136 страниц. Список литературы включает 77 наименований.

Краткое содержание работы

Во **введении** описываются цели работы, обосновывается ее актуальность и практическая значимость, перечисляются основные результаты.

Первая глава является вводной, в ней излагаются основные идеи и понятия, определяющие процесс планирования в системах автоматизированного динамического распараллеливания прикладных программ. В разделе 1.1 определяется задача планирования вычислений в системах динамического распараллеливания.

Отмечается, что традиционно под планированием понимается задача составления расписаний. Постановка этой задачи включает некоторые наборы работ и вычислительных узлов. Требуется определить порядок исполнения работ на этих узлах таким образом, чтобы достичь оптимального значения некоторой целевой функции. Важной особенностью этой задачи является тот факт, что вся информация о работах известна

заранее. Однако, во многих практически важных случаях планировщик не обладает такой информацией.

Далее в разделе 1.1 обращается внимание на тот факт, что в системах динамического распараллеливания в общем случае до запуска программы невозможно определить количество задач, которые будут созданы в процессе ее исполнения, порядок их создания, а также процессорное время, необходимое для их исполнения. Таким образом, в этих системах естественным образом возникает задача *динамического планирования*, также называемая задачей *балансировки нагрузки* (load balancing)[4].

В разделах 1.2 и 1.3 рассматриваются методы планирования в системах динамического распараллеливания. В разделе 1.4 приведен краткий обзор методов планирования в таких системах, как Cilk, CHARM++, GUM и T-система. Более подробно рассматривается T-система — подход к распараллеливанию программ, разрабатываемый в ИПС РАН и в МГУ имени М. В. Ломоносова[5]. Обращается внимание на тот факт, что существующие реализации T-системы, в том числе GRACE, OpenTS, NewTS, различаются как по своему внутреннему устройству, так и по возможностям, предоставляемым ими пользователю. Отмечается, что программный комплекс NewTS используется в качестве объекта для апробации результатов настоящей диссертационной работы.

В T-системе для разработки прикладных программ используется язык *TC*. Он представляет собой язык C++ с несколькими модификаторами, описывающими возможности параллельного исполнения отдельных ча-

[4] *Loidl H.-W.* Load balancing in a parallel graph reducer // Trends in functional programming. — Exeter, UK, UK: Intellect Books, 2002. — Pp. 63–74.

[5] Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии T-системы / С. М. Абрамов, В. А. Васенин, Е. Е. Мамчиц и др. // Тр. Всерос. науч. конф. «Высокопроизводительные вычисления и их приложения» (30 октября – 2 ноября 2000, г. Черноголовка). — М.: Изд-во МГУ, 2000. — С. 261–264.

T-система с открытой архитектурой / С. М. Абрамов, А. И. Адамович, А. В. Инюхин и др. // Тр. Междунар. науч. конф. «Суперкомпьютерные системы и их применение SSA'2004» (26–28 октября 2004, г. Минск). — Минск: ОИПИ НАН Беларуси, 2004. — С. 18–22.

стей программы. Основными модификаторами являются `tfun` и `tval`.

С помощью модификатора `tfun` объявляются так называемые *Т-функции*, которые могут исполняться на другом узле системы параллельно с вызывающей функцией. Вызов Т-функции не приводит к ее немедленному исполнению. Вместо этого вызывающая функция получает так называемое «неготовое» значение. После завершения выполнения функции значение станет «готовым» — содержащим результат вычислений.

Вызов Т-функции приводит к созданию *задачи* — объекта, описывающего ее отложенное исполнение. Задача может пересылаться между узлами вычислительной системы. *Планировщик* Т-системы определяет порядок исполнения задач и их пересылки между узлами с целью минимизации времени исполнения программы.

Алгоритмы планирования, используемые в версиях Т-системы, предшествующих NewTS, показывали свою неэффективность для рассматриваемого класса приложений (далее используется термин — Т-программа) даже для относительно небольшого ($\geqslant 50$) числа вычислительных узлов. Это обстоятельство, анализ механизмов планирования, существующих на настоящее время в системах динамического распараллеливания, используемых для этого алгоритмов, их сравнение с теми, которые реализованы в OpenTS, указывали на необходимость поиска новых подходов. В последующих главах диссертации рассмотрены возникающие на этом пути задачи и способы их решения, предлагаемые автором.

Во **второй главе** рассматривается разработанная автором математическая модель исполнения параллельных программ в системах динамического распараллеливания приложений. Описываются предложенные автором алгоритмы планирования, учитывающие специфику программного комплекса NewTS.

В разделе 2.1 отмечается, что при разработке планировщика для OpenTS, а также для ранних версий NewTS, целью ставилось достижение прием-

лемой эффективности на существовавших в то время приложениях для Т-системы. Математические методы для изучения механизмов планирования при этом не использовались. Недостатком такого подхода является то, что он ориентируется на конкретные классы приложений. Разработка новых прикладных программ для Т-системы часто выявляла недостатки в планировщике. Для их исправления в код планировщика (или других модулей системы) вносились изменения, которые приводили к более равномерной загрузке узлов на данной программе. При этом эффективность выполнения других программ, как правило, снижалась.

По этой причине возникает задача построения математической модели, описывающей процесс исполнения параллельной программы. Такая модель с одной стороны должна учитывать особенности Т-системы как объекта испытаний ее программной реализации. С другой стороны, она должна быть достаточно общей и применимой к широкому классу программ. Модель должна служить основой для формального описания и анализа алгоритмов планирования исполнения параллельных программ, способствовать лучшей интерпретации событий и предсказывать особенности поведения этих алгоритмов в различных условиях. Целью построения такой модели является:

- объяснение причин неэффективного исполнения некоторых прикладных программ в существующих средствах, основанных на балансировке нагрузки, и, в частности, в реализациях Т-системы;
- выделение класса прикладных программ, которые могут быть эффективно исполнены с помощью динамического планирования;
- создание алгоритма планирования, эффективно исполняющего программы упомянутого выше класса и формальное доказательство его эффективности.

В разделе 2.1 вводится предлагаемая автором математическая модель, описывающая процесс исполнения параллельных программ в систе-

ме динамического распараллеливания. При ее построении за основу была взята модель, изложенная в работах Р. Блюмоффа (Robert D. Blumofe) и Ч. Лейсерсона (Charles E. Leiserson)[6]. В ней параллельная программа представляется в виде ориентированного ациклического графа (V, E), где V — множество вершин графа, $E \subseteq V \times V$ — множество его ребер. Вершинам графа соответствуют *подзадачи*, участки последовательного исполнения программы. Выделяется начальная $v_s \in V$ вершина графа. Рассматриваются только программы, для которых V является конечным множеством, и любая вершина достижима из начальной.

С помощью ребер задается порядок исполнения задач. Согласно ему, подзадача начинает исполняться не раньше, чем завершатся все ее *предки* — вершины, из которых существует путь в данную. Выделяются три вида ребер, соответствующих операциям создания новой задачи, продолжения исполнения, и зависимости по данным между задачами.

Для описания временных характеристик процесса исполнения каждой подзадаче ставится в соответствие ее *вес*, или *вычислительная сложность*, то есть время, необходимое для ее исполнения. Вычислительная сложность задается с помощью функции $w : V \to \mathbb{R}_+$.

Определение 1. Набор $(V, v_s, E = E_c \sqcup E_s \sqcup E_d, w)$, удовлетворяющий описанным выше условиям, называется *параллельной программой*.

Вводится определение *плана исполнения* параллельной программы, который для каждой вершины $v \in V$ определяет время запуска и номер вычислительного узла[7]. Раздел 2.2 посвящен изучению свойств *жадных планов исполнения*, а именно — таких, в которых готовые к исполнению задачи начинают сразу исполняться при наличии свободных узлов.

[6] *Blumofe R. D., Leiserson C. E.* Scheduling multithreaded computations by work stealing // J. ACM. — 1999. — Vol. 46, no. 5. — Pp. 720–748.

[7] Под вычислительным узлом здесь и далее будем понимать один процессор или одно ядро в случае многоядерных процессоров в составе вычислительной системы.

Доказывается, что время исполнения параллельной программы на одном узле, обозначаемое T_1, а также время ее исполнения на неограниченном числе узлов — T_∞, не зависят от выбора плана, а являются характеристиками программы. Доказанная в разделе 2.2 теорема дает оценку эффективности жадных планов.

Теорема 1. *Пусть T_1 — общее время исполнения данной параллельной программы на одном узле, T_∞ — время исполнения программы на неограниченном количестве узлов. Тогда для любого жадного плана для P узлов справедливо неравенство $T_P \leqslant T_1/P + T_\infty$, где T_P — время исполнения программы в данном плане.*

Эта теорема является обобщением результатов Блюмоффа и Лейсерсона, которые доказали аналогичную теорему в рамках своей модели. Ключевое отличие между этими теоремами заключается в том, что в модели Блюмоффа и Лейсерсона веса всех вершин предполагаются равными единице, а в настоящей диссертации теорема доказана для программ с произвольными весами вершин. Доказательство заключается в выборе некоторого жадного плана для неограниченного числа узлов и его преобразовании в план для P узлов. Показывается, что при этом план остается жадным, а время исполнения возрастает не более, чем на T_1/P.

В разделе 2.3 описано обобщение модели на случай неоднородных вычислительных систем. Предполагается, что производительность (далее по тексту — мощность) каждого узла может быть описана одним числом.

Определение 2. *Вычислительным комплексом* называется конечная непустая последовательность $C = (c_1, c_2, \ldots, c_N), c_i \in \mathbb{R}_+, c_i \geqslant 0$. При этом величина $c_\Sigma = \sum_i c_i$ называется суммарной мощностью узлов, а величина $c_{min} = \min_i c_i$ — минимальной мощностью узлов.

В этом случае вычислительная сложность задачи обозначает не время ее исполнения, а число операций, необходимое для ее завершения. Время исполнения подзадачи v на узле i считается равным $w(v)/c_i$. В этих

условиях определения T_1 и T_∞ становятся некорректными. Вместо них используются следующие определения.

Определение 3. *Общей вычислительной сложностью* параллельной программы (V, v_s, E, w) называется сумма весов ее вершин, $W_1 = \sum_{v \in V} w(v)$.

Определение 4. *Глубиной* параллельной программы (V, v_s, E, w) называется максимум суммы весов вершин во всех путях в графе (V, E):

$$W_\infty = \max_{(v_1, v_2, \ldots, v_k) \text{ — путь в } (V,E)} \sum_{i \in [1,k]} w(v_i).$$

В отличие от T_1 и T_∞, которые задают время исполнения параллельной программы, W_1 и W_∞ показывают ее вычислительную сложность и измеряются в операциях.

С использованием приведенных выше обозначений, следующая теорема обобщает теорему 1 и дает оценку времени исполнения параллельной программы с использованием жадных планов в условиях различной вычислительной мощности узлов.

Теорема 2. *Пусть $C = (c_1, c_2, \ldots, c_P)$ — вычислительный комплекс. Для любого жадного плана на этом комплексе справедливо неравенство*

$$T_C \leqslant \frac{W_1}{c_\Sigma} + \frac{W_\infty}{c_{min}}$$

где T_C — время исполнения программы в данном плане.

С использованием теоремы 2 коэффициент эффективности параллельной программы оценивается как

$$K_{eff} = \frac{W_1}{T_C c_\Sigma} \geqslant \frac{W_1}{\left(\frac{W_1}{c_\Sigma} + \frac{W_\infty}{c_{min}}\right) c_\Sigma} = \frac{1}{1 + \frac{W_\infty}{W_1} \cdot \frac{c_\Sigma}{c_{min}}},$$

и оценка является асимптотически оптимальной при $W_1/W_\infty \gg c_\Sigma/c_{min}$. Таким образом, теоремы 1 и 2 показывают, что при изучении алгоритмов

планирования исполнения параллельных программ можно ограничиться алгоритмами, действующими жадным образом.

В разделе 2.4 приводится обобщение модели на случай распределенных вычислительных систем. Предполагается, что время, необходимое для обращения к произвольному удаленному узлу и получению ответа, фиксировано. Такое время обозначается через t_{RTT}.

В рамках принятого предположения в разделе 2.4 задается и подробно описывается вероятностный алгоритм планирования, действующий по методу заимствования заданий. Для программ, работающих под управлением этого алгоритма, верна следующая теорема, оценивающая время исполнения на P одинаковых узлах.

Теорема 3. *Пусть (V, v_s, E, w) — параллельная программа, в которой каждая вершина имеет не более двух исходящих ребер. Пусть t_{RTT} — время обращения за задачей к произвольному узлу, T_1 — время исполнения программы на одном узле, T_∞ — время ее исполнения на неограниченном числе узлов, t_{min} — минимальное время исполнения подзадачи из V. Тогда математическое ожидание общего времени исполнения программы на P узлах с использованием указанного выше алгоритма оценивается следующим образом:*

$$E(T_P) < 2\frac{T_1}{P} + \left(2 + \frac{(3 + 2\ln P)\, t_{RTT}}{t_{min}}\right) T_\infty + t_{min}.$$

В доказательстве этой теоремы рассматриваются два случая: когда используется более $P/2$ узлов, и когда более половины узлов простаивают. В первом случае оценивается выполняемая вычислительная работа, и делается вывод, что такое состояние может длиться не более $2T_1/P$ времени. Во втором случае оценивается время поиска готовых к исполнению задач свободными узлами.

В разделах 2.5 и 2.6 описываются два алгоритма планирования для системы NewTS. Алгоритм Fishing действует по методу заимствования

заданий. Узел, на котором отсутствуют готовые к исполнению задачи, отправляет запрос на другой, случайным образом выбранный узел. В каждый момент времени с одного узла может быть отправлен только один запрос, который может пересылаться случайным образом до тех пор, пока не найдет узел с готовой к исполнению задачей. Максимальное число пересылок ограничено параметром планировщика. Описаны две модификации этого алгоритма, повышающие производительность в случае программ с централизованным порождением задач, а также на распределенных вычислительных установках.

Алгоритм балансирующего планировщика действует по методу разделения работы. Узлы под управлением этого планировщика обмениваются информацией о количестве готовых к исполнению задач. Пересылки осуществляются таким образом, чтобы достичь максимально равномерного распределения задач. Упомянутые алгоритмы реализованы в планировщиках для системы NewTS. Они имеют несколько параметров, влияние которых на исполнение пользовательских программ изучается в главе 5.

Описанные в разделах 2.5 и 2.6 алгоритмы планирования достигают высокой эффективности планирования для многих приложений. Однако, для этих алгоритмов отсутствуют оценки эффективности. В связи с этим обстоятельством, автором был реализован в программном комплексе NewTS *неблокирующий* режим исполнения, описанный в разделе 2.7.

В неблокирующем режиме используется алгоритм планирования, используемый в доказательстве теоремы 3. По этой причине, для программ, исполняемых в этом режиме, справедлива оценка из теоремы 3. Математическое ожидание времени исполнения программ в этом режиме на P узлах убывает как $1/P$ при условии $T_1/T_\infty \geqslant P \ln P$.

Третья глава диссертации посвящена разработке и реализации методов эффективного исполнения *мелкозернистых*[8] (fine-grained) парал-

[8] *Andersen B.* Fine-grained parallelism in Ellie // *J. Object Oriented Program.* — 1992. — Vol. 5, no. 3.

лельных программ. При исполнении таких программ создается большое число T-функций, каждая из которых имеет относительно малую вычислительную сложность. Процессорное время, расходуемое при этом на создание, удаление и планирование задач, а также на создание вспомогательных объектов, таких как T-переменные и заменители, может превысить время полезной работы. Применение методов планирования, изложенных в главе 2, к мелкозернистым программам оказывается неэффективным.

В диссертации рассматривается метод эффективного исполнения мелкозернистых программ, основанный на объединении нескольких задач в одну во время исполнения программы. Такое объединение в литературе называется *включением задач* (task inlining). Решение о возможности объединения принимается на основе информации о загруженности узлов вычислительной системы. Включение задач ускоряет исполнение мелкозернистых программ за счет уменьшения накладных расходов на создание и использование структур ядра NewTS.

В разделе 3.1 рассматривается пример мелкозернистой параллельной программы. В разделе 3.2 приводится краткий обзор методов включения задач. По результатам обзора для применения в NewTS выбирается метод включения задач, основанный на загруженности вычислительных узлов, также называемый *load-based inlining*.

В разделах 3.3 и 3.4 описываются детали реализации метода включения задач в NewTS. Особое внимание уделяется корректности этой операции. В некоторых случаях включение задач может изменить результат работы программы по той причине, что дочерняя задача использует контекст родительской, и переключение на нее невозможно. Таким образом,

— Pp. 55–62.

Blelloch G. E., Gibbons P. B., Matias Y. Provably efficient scheduling for languages with fine-grained parallelism // SPAA '95: Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures. — New York, NY, USA: ACM Press, 1995. — Pp. 1–12.

программа может оказаться в состоянии *взаимоблокировки* (deadlock).

В диссертации предлагается достаточное условие корректности, состоящее в том, что включение некоторой функции является *безопасным* (не может привести к взаимоблокировке), если все ее предки не используют *отложенных* значений. Разделы 3.3 и 3.4 посвящены методам эффективной проверки этого условия.

В **четвертой главе** излагаются особенности программной реализации планировщиков в системе NewTS. В разделе 4.1 описана структура данных для хранения готовых к исполнению задач. Модуль планирования NewTS позволяет определить несколько планировщиков и выбирать планировщик, используемый при запуске программы. В разделах 4.2 и 4.3 описаны интерфейсы модуля планирования и отдельных планировщиков. В разделах 4.4 и 4.5 приведены детали реализации балансирующего планировщика и планировщика Fishing.

Пятая глава посвящена тестированию разработанных планировщиков, которое проводилось на вычислительном кластере МВС-15000 ВМ МСЦ РАН, а также на гетерогенной распределенной вычислительной системе, описанной далее. Кластер МВС-15000 состоит из 574 узлов, каждый из которых содержит два процессора PowerPC 970 2.2 ГГц. Для передачи данных используется коммуникационная сеть Myrinet.

Одна из программ, используемых в тестировании — программа RT, которая строит изображение трехмерных объектов с высоким разрешением методом трассировки лучей. Граф вызовов в программе RT представляет собой сбалансированное в смысле числа вершин бинарное дерево. Каждая листовая T-функция вычисляет прямоугольную область изображения, их вычислительная сложность различна и зависит от входных данных.

На рис. 1 показаны результаты измерений эффективности распараллеливания программы RT. Использовались различные алгоритмы пла-

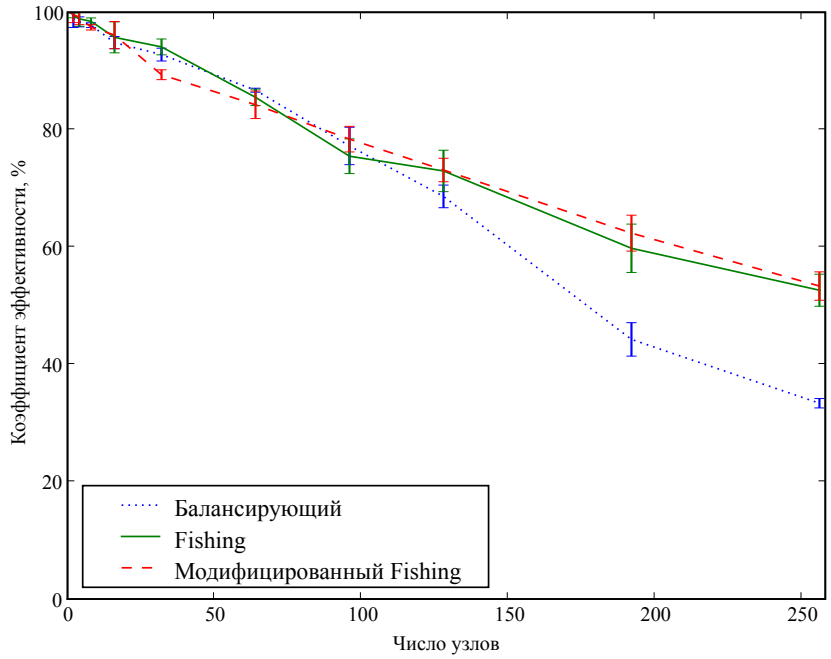

Рис. 1: Эффективность работы программы RT.

нирования. Коэффициент эффективности (КЭ) распараллеливания программы на N узлах определяется по формуле $\text{КЭ}_N = T_1/(N \cdot T_N)$.

Программа prgdemo моделирует поведение программного комплекса Vortex, предназначенного для моделирования двумерного нестационарного обтекания твердых тел потоком несжимаемой среды с использованием метода дискретных вихрей и метода вязких вихревых доменов. Программа prgdemo обладает параллелизмом типа «scatter-gather», когда на несколько узлов рассылаются входные данные, которые затем обрабатываются и результаты собираются на исходном узле. Похожие алгоритмы используются во многих итеративных задачах. Результаты измерений эффективности распараллеливания prgdemo показаны на рис. 2.

Планировщик Fishing имеет ряд параметров. В разделе 5.6 изучается влияние этих параметров на исполнение параллельных программ и выбираются их оптимальные значения в определенных условиях.

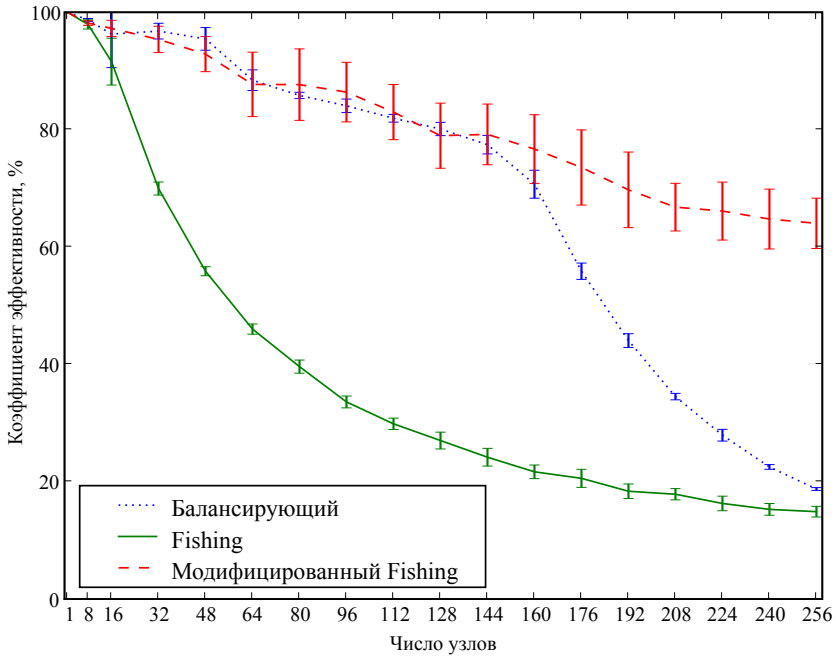

Рис. 2: Эффективность работы модельной программы prgdemo.

Произведено две серии экспериментов: в первом число создаваемых для построения изображения задач составляло 2^{21}, во втором — 2^{11}. Таким образом, исследовалось поведение планировщика Fishing при различных значениях параметров как на крупных, так и на мелких задачах. Все запуски производились с использованием 64 узлов. Параметр ttl принимал значения от 1 до 24 (28 в случае jobsize=1024), параметр delay — значения 0.001, 0.01, 0.1, 0.5 и 1.0 секунды. Результаты измерений приведены в тексте диссертации.

Исходя из результатов измерений, в качестве значений по умолчанию для планировщика Fishing были выбраны значения ttl=10 и delay=0.01 секунды. Отмечено также, что при запуске на коммуникационной сети с низкой пропускной способностью, значение delay может быть увеличено до 0.1 секунды с целью уменьшения нагрузки на сеть.

В последние годы наблюдается активное развитие вычислительный

систем, построенных по технологии GRID. В этой связи, особый интерес представляет исследование разработанных планировщиков в условиях распределенной гетерогенной вычислительной среды.

Измерения проводились на неоднородной распределенной системе, состоящей из двух вычислительных кластеров, являющихся частью GRID полигона, рассредоточенного на сети МГУ-РАН. Один из таких кластеров расположен в НИИ Механики МГУ и состоит из 8 двухпроцессорных узлов AMD Athlon MP 1800+. В качестве коммуникационной среды используется SCI. Второй кластер НКС-160 установлен в Суперкомпьютерном центре ВМиМГ СО РАН в Новосибирске. Он состоит из 80 вычислительных модулей HP Integrity rx1620, каждый из которых содержит два процессора Intel Itanium 2 1.6 ГГц. Узлы связаны посредством высокопроизводительной коммуникационной сети InfiniBand. Для связи между кластерами использовалась виртуальная сеть, организованная через Интернет. Ее пропускная способность составляла 1.2 Мбайт/с, а задержка доставки пакетов — 50 мс.

Измерения производились на следующих программах:

- тестовая программа fib, вычисляющая значения чисел Фибоначчи рекурсивным методом, с параметром 42;
- программа RT, со сценой, содержащей 820 объектов, и разрешением выходного изображения 2048×2048 пикселов.

Для оценки теоретически достижимого времени исполнения программы на двух кластерах использовалась формула $T_{\text{теор}} = (1/T_1 + 1/T_2)^{-1}$, где T_i — время исполнения программы на кластерах, а $T_{\text{теор}}$ — время ее исполнения при максимальной утилизации вычислительных ресурсов двух кластеров. Коэффициент эффективности (КЭ) определялся как отношение $T_{\text{теор}}$ к наблюдаемому времени исполнения.

Табл. 1 и 2 содержат результаты измерений времени исполнения программы на отдельных кластерах, при совместном запуске, а также оцен-

Программа	T_1, с	T_2, с	Теоретически достижимое время, с	T_{1+2}, с	КЭ, %
fib(42)	130.73	126.28	64.23	67.00	95.9
RT, 2048×2048	68.84	60.58	32.22	35.57	90.6

Таблица 1: Время исполнения программы на 16 + 26 процессорах.

Программа	T_1, с	T_2, с	Теоретически достижимое время, с	T_{1+2}, с	КЭ, %
fib(42)	130.73	33.51	26.67	31.08	85.8
RT, 2048×2048	68.84	18.33	14.48	17.80	81.3

Таблица 2: Время исполнения программы на 16 + 100 процессорах.

ку идеального времени исполнения согласно указанной выше формуле. Можно отметить, что при при вычислениях на 16+26 процессорах реальное время исполнения близко к теоретически достижимому, что свидетельствует об эффективном распределении работы планировщиком. Программа RT производит большое число пересылок (на 16+26 процессорах общий объем пересылаемых данных составил 47 Мбайт), что приводит к меньшему, чем у программы fib, коэффициенту эффективности.

В разделе 5.9 отмечено, что были произведены тестовые запуски на модельных и практически важных прикладных задачах, а также исследования влияния различных параметров планировщиков на эффективность исполнения программ. Испытания на неоднородных распределенных вычислительных установках показали, что разработанные планировщики достигают эффективность исполнения, близкую к максимальной.

Основные результаты работы

В диссертационной работе получены следующие основные результаты:

- разработана математическая модель, представляющая параллельную программу в виде ориентированного ациклического графа подзадач и описывающая процесс исполнения такой программы в рас-

пределенных системах и в системах с различной производительностью узлов; получены оценки времени исполнения параллельных программ в таких системах;

- разработаны и программно реализованы с использованием системы автоматизированного динамического распараллеливания NewTS два алгоритма планирования, основанные на методах балансировки нагрузки и заимствования заданий, уменьшающие нагрузку на коммуникационную сеть и общее время исполнения вычислительных приложений, в том числе в распределенных системах;
- разработан и реализован в системе NewTS механизм исполнения порождаемых задач, основанный на анализе загруженности узлов, снижающий системные накладные расходы при исполнении мелкозернистых параллельных программ.

Публикации по теме диссертации

Основные результаты работы изложены в следующих публикациях.

1. *Степанов Е. А.* Метапланирование в открытой T-системе: механизмы, модели и инструментальные средства // Тез. докл. науч. конф. «Ломоносовские чтения» (18–28 апреля, МГУ им. Ломоносова, Москва). — М.: Изд-во Московского университета, 2005. — С. 174.
2. *Степанов Е. А.* Планирование в OpenTS — системе автоматического динамического распараллеливания // Информационные технологии и программирование: Межвузовский сборник статей. Под ред. В. А. Васенина, Д. Л. Ревизникова, Е. А. Роганова. Вып. 2 (14). — М.: МГИУ, 2005. — С. 31–42.
3. *Степанов Е. А.* Автоматический выбор гранулы параллелизма в системе автоматического динамического распараллеливания // Тез. докл. науч. конф. «Ломоносовские чтения» (17–27 апреля, МГУ

им. Ломоносова, Москва). — М.: Изд-во Московского университета, 2006. — С. 135–136.

4. *Степанов Е. А.* Новые механизмы планирования в системе автоматического динамического распараллеливания программ // Тезисы III Междунар. конф. по проблемам управления МКПУ-2006 (20–22 июня 2006, ИПУ РАН, Москва). — М.: Ин-т проблем управления, 2006. — С. 138.
5. *Конев И. М., Степанов Е. А.* Автоматизация динамического распараллеливания программ: планирование, управление памятью, работа в гетерогенной среде // *Информационные технологии.* — 2007. — № 10. — С. 71–73.
6. *Степанов Е. А.* Математическая модель планирования в системах автоматизированного динамического распараллеливания // Тез. докл. науч. конф. «Ломоносовские чтения» (16–25 апреля, МГУ им. М. В. Ломоносова, Москва). — М.: Изд-во Московского университета, 2007. — С. 141.
7. *Конев И. М., Степанов Е. А.* Автоматизация динамического распараллеливания программ: планирование, управление памятью, работа в гетерогенной среде // Приложение к журналу «Информационные технологии». — № 10. М.: Изд-во «Новые технологии», 2007. — 32 с.

В работе 7 автору настоящей диссертации принадлежат разделы «Введение», «Планирование исполнения программ» и «Включение задач».